vwr-package

Useful functions for visual word recognition research

Description

Functions and data for use in visual word recognition research: Supports computation of neighbors using Hamming, Levenshtein, and Restricted Levenshtein-Damerau distances, average distances to neighbors (e.g., OLD20), and Coltheart’s N. Supplies the LDkNN algorithm to detect bias in the composition of a lexical decision task. Most of the functions support parallel execution. Supplies wordlists for several languages.

Details

Package: vwr
Type: Package
Version: 0.3
Date: 2013-08-13
License: GPL-3

Author(s)

Emmanuel Keuleers
Maintainer: emmanuel.keuleers@ugent.be
Compute average Levenshtein distances

Description

Compute the average Levenshtein distances between a word and its n nearest neighbors in a lexicon.

Usage

ald(sourcesL targetsL nL method"levenshtein", parallel = FALSE)
old20(sourcesL targetsL method="levenshtein", parallel = FALSE)

Arguments

- sources: a list of words for which the average Levenshtein distance should be computed. Must be of type character, or convertible to type character with as.character.
- targets: a list of words containing possible neighbors. Must be of type character, or convertible to type character with as.character.
- method: specifies the distance function. With "levenshtein", levenshtein.distance is used, with "levenshtein.damerau" levenshtein.damerau is used.
- n: specifies the number of nearest neighbors on which the average should be based. The variant old20 does not take the n argument (it is fixed to 20).
- parallel: with parallel=TRUE, ald will run in parallel an multiple cores. The number of parallel processes is specified by detectCores(logical = FALSE).

Details

The OLD20 measure was originally proposed by Yarkoni et al. (2008). This implementation is orders of magnitude faster than Tal Yarkoni’s LDcalc program (see http://talyarkoni.com/materials.php). Do not use multicore=TRUE in a GUI environment, as it will most likely crash your R session.

Value

A vector of average Levenshtein distances with names corresponding to sources.

Author(s)

Emmanuel Keuleers

References

See Also

levenshtein.distance, levenshtein.neighbors

Examples

```r
data(basque.words)
ald(basque.words[1:10], basque.words, 20)
old20(basque.words[1:10], basque.words)
```

---

basque.words  

*A list of Basque Words*

---

Description

A list of 18,483 Basque words without spaces or dashes, from E-HITZ.

Usage

```r
data(basque.words)
```

Format

A character vector.

Source


Examples

```r
data(basque.words)
```

---

coltheart.N  

*Compute Coltheart’s N*

---

Description

Compute Coltheart’s N measure (the number of neighbors at distance 1).

Usage

```r
coltheart.N(sources, targets, distance = 1, method = "hamming", parallel = FALSE)
```
Arguments

sources  
a list of words for which Coltheart’s N should be computed. Must be of type character, or convertible to type character with as.character.

targets  
a list of words containing possible neighbors. Must be of type character, or convertible to type character with as.character.

parallel  
with parallel=TRUE, coltheart.N will run in parallel an multiple cores. The number of parallel processes is specified by detectCores(logical = FALSE).

distance  
specifies the distance on which N should be based. This should be left to 1 to compute the original measure.

method  
with method="hamming", compute N based on the hamming.distance, with method="levenshtein", compute N based on the levenshtein.distance with method="levenshtein-damerau", compute N based on the levenshtein.damerau.distance

Value

An integer vector with names corresponding to sources.

Author(s)

Emmanuel Keuleers

References


See Also

hamming.distance, levenshtein.distance

Examples

data(spanish.words)
sample.words<-sample(spanish.words,20)
coltheart.N(sample.words,spanish.words)
coltheart.N(sample.words,spanish.words, method='levenshtein')


dutch.words       A list of Dutch Words

Description

A list of 293,749 Dutch words without spaces or dashes from the CELEX lexical database.

Usage

data(dutch.words)
Format

A character vector.

Source


Examples

data(english.words)

---

english.words | A list of English Words

---

Description

A list of 66,330 English words without spaces or dashes from the CELEX lexical database.

Usage

data(english.words)

Format

A character vector.

Source


Examples

data(english.words)
A list of 116,194 French words without spaces or dashes from the Lexique 3 database.

Usage

data(french.words)

Format

A character vector.

Source

http://www.lexique.org/

Examples

data(french.words)

german.words  A list of German Words

Description

A list of 315,391 German words without spaces or dashes from the CELEX lexical database.

Usage

data(german.words)

Format

A character vector.

Source


Examples

data(german.words)
hamming.distance  
Compute Hamming distances

Description
Compute the Hamming distance (the number of non-overlapping characters) between words of the same length.

Usage
hamming.distance(xsource, targets)

Arguments
- xsource: A character string to compute the Hamming distance from.
- targets: Words to which the Hamming distance must be computed. Must be of type character, or convertible to type character with as.character.

Details
The actual distance computation is performed by stringdist with "method='h'".

Value
An integer vector containing Hamming distances, with names corresponding to targets. Since the Hamming distances is only defined between words of the same length, the output of hamming.distance is only guaranteed to have the same length as targets if all targets have the same length as source.

Author(s)
Emmanuel Keuleers

References

See Also
stringdist, hamming.neighbors,coltheart.N

Examples
```r
data(english.words)
targets<-english.words[which(nchar(english.words)==5)]
hamming.distance('electroencephalogram', english.words)
```
hamming.neighbors

Compute Hamming neighbors

Description

List the neighbors of a character string by Hamming distance.

Usage

hamming.neighbors(source, targets)

Arguments

source

A character string.

targets

Potential Hamming neighbors.

Value

A list of neighbors at each distance.

Author(s)

Emmanuel Keuleers

See Also

hamming.distance, stringdist

Examples

data(english.words)
hamming.neighbors('electroencephalogram', english.words)
hamming.neighbors('hello', english.words)

ldknn

Run the ldknn algorithm

Description

The ldknn algorithm is used to detect bias in the composition of a lexical decision task, using k-nearest neighbor classification and the Levenshtein distance metric.

Usage

ldknn(stimuli, types, reference, k = 1, method='levenshtein', parallel = FALSE)
**Arguments**

- **stimuli**: character strings corresponding to the stimuli in the experiment.
- **types**: factor corresponding to the type of each stimulus in the experiment.
- **reference**: a character string giving the reference level. Must be a level of the factor in `types`.
- **k**: a value for the k parameter. Set to 1 by default.
- **method**: 
  - "levenshtein": uses `levenshtein.distance` to calculate distances
  - 'levenshtein.damerau': uses `levenshtein.damerau.distance` to calculate distances
- **parallel**: with parallel=TRUE, `ldknn` will run in parallel an multiple cores. The number of parallel processes is specified by `detectCores(logical = FALSE)`.

**Details**

Combining k nearest neighbor classification with the Levenshtein distance produces an algorithm which can be described as follows. For an experiment containing a number of stimuli, which can be words or nonwords:

1. Compute the Levenshtein distances between the currently presented stimulus and all previously presented stimuli.
2. Identify the previously presented stimuli that are at the k nearest distances from the current stimulus.
3. Compute the probability of a word response for the given stimulus based on the relative frequency of words among the nearest neighbors.

**Value**

A list with class `ldknn.run`.

- **data**: A data frame containing the results of the run. stimulus gives the stimulus values, type gives the types of the stimuli, p gives the probability for a reference level response for that stimulus.
- **reference level**: The reference level used for the simulation.
- **Odds**: The odds, z value, and p value for a reference level response, resulting from a logistic regression in which the probabilities generated by the `ldknn` algorithm are used to predict stimulus types.

`plot` and `print` methods are available for objects of class `ldknn.run`

**Author(s)**

Emmanuel Keuleers

**References**

ldknn.odds

See Also

levenshtein.distance, levenshtein.damerau.distance

Examples

data(english.words)
data(basque.words)
# set up a mock experiment: English stimuli are words, Basque stimuli are nonwords
experiment<-data.frame(stimulus=c(sample(english.words,500), sample(basque.words,500)),
  type=factor(rep(c('Word','Nonword'),each=500),levels=c('Word','Nonword')))
# randomize the trials
experiment<-experiment[sample(1:1000,1000),]
# run the ldknn algorithm
results<-ldknn(experiment$stimulus,experiment$type,'Word')
print(results)
plot(results)

ldknn.odds Compute the odds of correctly predicting a response

Description

Perform a logistic regression to compute the odds of correctly predicting a particular response. Used internally by ldknn.run. Not intended for separate use

Usage

ldknn.odds(type, probability, reference)

Arguments

  type          A factor corresponding to the true class for each response.
  probability   Probability of the reference level response.
  reference     A character string corresponding to the reference level.

Author(s)

  Emmanuel Keuleers

See Also

  ldknn
**Description**

Compute the Levenshtein-Damerau distance between two character strings (the minimal number of insertions, deletions replacements, or transpositions required to transform one string into the other).

**Usage**

```r
levenshtein.damerau.distance(xsource, targets)
```

**Arguments**

- `xsource`: A character string to compute the Levenshtein-Damerau distance from.
- `targets`: A list of words to compute the Levenshtein-Damerau distance to. Must be of type character, or convertible to type character with as.character.

**Details**

The distance computation is performed by `stringdist` with method="osa". Note that this function computes the restricted Levenshtein-Damerau distance instead of the unrestricted version.

**Value**

An integer vector containing Restricted Levenshtein-Damerau distances, with names corresponding to targets.

**Author(s)**

Emmanuel Keuleers

**References**


**See Also**

`levenshtein.neighbors`, `levenshtein.distance`, `stringdist`, `ald`

**Examples**

```r
data(french.words)
levenshtein.damerau.distance('pourquoi', sample(french.words, 20))
```
levenshtein.damerau.neighbors

Compute Levenshtein-Damerau neighbors

**Description**

List the neighbors of a character string by Levenshtein-Damerau distance.

**Usage**

levenshtein.damerau.neighbors(xsource, targets)

**Arguments**

- `xsource`: A character string.
- `targets`: Potential neighbors. Must be of type character, or convertible to type character with `as.character`.

**Value**

A list of neighbors at each distance.

**Author(s)**

Emmanuel Keuleers

**See Also**

levenshtein.damerau.distance

**Examples**

data(serbian_latin.words)
levenshtein.neighbors('pola', serbian_latin.words)[1:2]

---

levenshtein.distance

Compute Levenshtein distances

**Description**

Compute the Levenshtein distance between two character strings (the minimal number of insertions, deletions or replacements required to transform one string into the other)

**Usage**

levenshtein.distance(xsource, targets)
Arguments

- `xsource`: A character string to compute the Levenshtein distance from.
- `targets`: A list of words to compute the Levenshtein distance to. Must be of type `character`, or convertible to type `character` with `as.character`.

Details

The distance computation is performed by `stringdist` with method="lv".

Value

An integer vector containing Levenshtein distances, with names corresponding to `targets`.

Author(s)

Emmanuel Keuleers

References


See Also

`levenshtein.neighbors`, `stringdist`, `ald`, `levenshtein.damerau.distance`

Examples

```r
data(french.words)
levenshtein.distance('pourquoi', sample(french.words, 20))
```

Description

List the neighbors of a character string by Levenshtein distance.

Usage

`levenshtein.neighbors(xsource, targets)`

Arguments

- `xsource`: A character string.
- `targets`: Potential Levenshtein neighbors. Must be of type `character`, or convertible to type `character` with `as.character`. 
### Value

A list of neighbors at each distance.

### Author(s)

Emmanuel Keuleers

### See Also

- `levenshtein.distance`

### Examples

```r
data(serbian_latin.words)
levenshtein.neighbors('pola', serbian_latin.words)[1:2]
```

### Description

See `ald`

### plot.ldknn.run

*Plot the results of an ldknn run*

### Description

See `ldknn`

### print.ldknn.run

*Print the results of an ldknn run*

### Description

See `ldknn`
serbian_cyrillic.words

A list of Serbian Words in Cyrillic alphabet

Description

A list of 144,105 words without spaces or dashes from the frequency dictionary of contemporary Serbian language, in Cyrillic alphabet.

Usage

data(serbian_cyrillic.words)

Format

A character vector.

Source


Examples

data(serbian_cyrillic.words)

---

serbian_latin.words

A list of Serbian Words in Latin alphabet

Description

A list of 144,105 words without spaces or dashes from the frequency dictionary of contemporary Serbian language, in Latin alphabet.

Usage

data(serbian_latin.words)

Format

A character vector.
Source


Examples

data(serbian_latin.words)

data(spanish.words)

Description

A list of 31,490 Spanish words without spaces or dashes, from the base-lexicon of BuscaPalabras.

Usage

data(spanish.words)

Format

A character vector.

Source


Examples

data(spanish.words)

data(vietnamese.words)

Description

A list of 47,966 Vietnamese words without spaces or dashes.

Usage

data(vietnamese.words)
Format

A character vector.

Examples

data(vietnamese.words)
Index

*Topic package
  vwr-package, 2

ald, 3, 12, 14, 15
basque.words, 4
coltheart.N, 4, 8
dutch.words, 5
english.words, 6
french.words, 7
german.words, 7
hamming.distance, 5, 8, 9
hamming.neighbors, 8, 9
ld1nn (ldknn), 9
ldknn, 9, 11, 15
ldknn.odds, 11
levenshtein.damerau.distance, 5, 10, 11, 12, 13, 14
levenshtein.damerau.neighbors, 13
levenshtein.distance, 4, 5, 10–12, 13, 15
levenshtein.neighbors, 4, 12, 14, 14
old20, 15
old20 (ald), 3
plot.ldknn.run, 15
print.ldknn.run, 15

serbian_cyrillic.words, 16
serbian_latin.words, 16
spanish.words, 17
stringdist, 8, 9, 12, 14

vietnamese.words, 17
vwr (vwr-package), 2
vwr-package, 2